Fast morphological attribute operations using Tarjan's union-find algorithm

Wilkinson, Michael H.F.; Roerdink, Jos B.T.M.

Published in:
MATHEMATICAL MORPHOLOGY AND ITS APPLICATIONS TO IMAGE AND SIGNAL PROCESSING

FAST MORPHOLOGICAL ATTRIBUTE OPERATIONS USING TARJAN'S UNION-FIND ALGORITHM

MICHAEL H. F. WILKINSON and JOS B. T. M. ROERDINK
Institute for Mathematics and Computing Science,
University of Groningen, P.O. Box 800, 9700 AV, Groningen,
The Netherlands

Abstract. Morphological attribute openings and closings and related operators are generalizations of the area opening and closing, and allow filtering of images based on a wide variety of shape or size based criteria. A fast union-find algorithm for the computation of these operators is presented in this paper. The new algorithm has a worst case time complexity of $O(N \log N)$ where N is the image size, as opposed to $O(N^2 \log N)$ for the existing algorithm. Memory requirements are $O(N)$ for both algorithms.

Key words: Area Operators, Attribute Operators, Granulometries, Union-find Algorithm.

1. Introduction

Morphological attribute openings, thinnings and granulometries were introduced by Breen and Jones [1] as a generalization of morphological area operators proposed by Vincent [8, 9]. Attribute openings are most easily understood in the binary case. Unlike structural openings, attribute openings are shape preserving, because they simply test whether a connected component satisfies some increasing criterion T. If it does, it is retained, if not, it is removed. In the case of the area opening, the area of each component is compared to some threshold value λ, and if the area of the component is larger, it is retained. The flexibility of this methodology is shown in Figure 1. In this figure a binary image of bacteria is filtered using three attribute openings, each of which would remove all squares smaller than 11 x 11 pixels. The first is the area opening, with $\lambda = 121$. All small bacteria have been removed in the resulting image. By contrast, the attribute opening using the criterion that the moment of inertia I must be larger than $\lambda = 11^4/6$, removes most of the smaller components, but not the elongated ones. Attribute opening using the length of the diagonal of the minimum enclosing rectangle as criterion, with $\lambda = \sqrt{242}$, has similar results.

The algorithm Breen and Jones derive for their wider class of operators is based on Vincent's pixel queue algorithm for area operators. Recently, a new algorithm for area openings and closings has been developed [5], which is based on Tarjan's union-find algorithm [7]. It was found that the union-find based algorithm was between 2 and 10 times faster than the original algorithm on the images tested. Furthermore, the computational burden of the new algorithm was practically independent of the size criterion λ used, or the image content. By contrast, Vincent's algorithm is particularly sensitive to the presence of

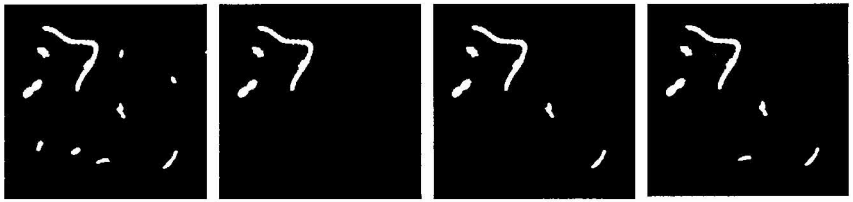

Fig. 1. Attribute openings of an image of bacteria: (a) a binary image of 256 x 256 pixels; attribute using (b) area $A \geq 121$; (c) Moment of inertia $I \geq 1\,1^4/6$, corresponding to that of an 11 × 11 square, and (d) length of diagonal of minimum enclosing rectangle $D \geq \sqrt{242}$. Structural opening of (a) by an 11 × 11 square structuring element removes all objects.

linear structures in the image, in which case the computing time rises almost linearly with λ.

In this paper we extend the union-find algorithm to the wider class of attribute openings and closings. Later work will focus on extension to thinnings and thickenings, and granulometries or size distributions.

2. Attribute Morphology: Theory

The theory of attribute operators is given only briefly here. For a more thorough discussion the reader is referred to [1]. Here we will first discuss binary attribute openings and closings, and then the extension to the grey scale case. Binary attribute openings are based on binary connected openings. Let the set $X \subseteq \mathbf{M}$ denote a binary image with domain $\mathbf{M}$. The binary connected opening $\Gamma_x(X)$ of X at point $x \in \mathbf{M}$ yields the connected component of X containing x if $x \in X$, and $\emptyset$ otherwise. Thus Γ_x extracts the connected component to which x belongs, discarding all others. Breen and Jones then use the concept of trivial openings Γ_T, which use an increasing criterion T to accept or reject connected sets. A criterion T is increasing if the fact that C satisfies T implies that D satisfies T for all $D \supseteq C$. The trivial opening Γ_T of a connected set C with increasing criterion T is just the set C if C satisfies T, and is empty otherwise. Furthermore, $\Gamma_T(\emptyset) = \emptyset$. The binary attribute opening is defined as follows.

Definition 1 *The binary attribute opening* Γ^T *of set* X *with increasing criterion* T *is given by*

$$\Gamma^T(X) = \bigcup_{x \in X} \Gamma_T(\Gamma_x(X)) \tag{1}$$

It can be shown that this is an opening because it is increasing, idempotent, and anti-extensive [1]. Theattribute opening is equivalent to performing a trivial opening on all connected components in the image.

A generalization to grey scale can be made by first defining thresholded images $X_h(f)$,

$$X_h(f) = \{x \in \mathbf{M} | f(x) \geq h\} \tag{2}$$

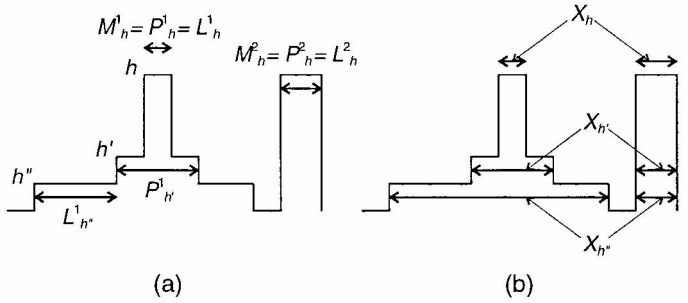

Fig. 2. One dimensional discrete image with grey levels $h > h' > h''$ to illustrate the definitions of level components, regional maxima, peak components, and the threshold images: (a) double arrows indicate three level components L^1_h, L^2_h and $L^1_{h''}$; the former two are also both peak components P^1_h and P^2_h and regional maxima at level h; a further peak component $P^1_{h'}$ at level h' is also shown; (b) shows the threshold sets $X_h, X_{h'}$, and $X_{h''}$ in relationship to the grey scale image.

where the grey scale image f is a mapping from the image domain $\mathbf{M}$ to $\mathbb{Z} \cup \{-\infty, \infty\}$.

Definition 2 *The grey scale attribute opening* γ^T *of image* f *with increasing criterion* T *is given by*

$$(\gamma^T(f))(x) = \max\{h | x \in \Gamma^T(X_h(f))\}. \tag{3}$$

Grey scale attribute closings can easily be defined by a duality relationship with the grey scale attribute openings [5].

3. Algorithms

Before going into the details of the algorithms, we first define a *level component* L_h at level h of a grey scale image f as a connected component of the set of pixels $\{p \in \mathbf{M} | f(p) = h\}$. A *regional maximum* M_h at level h is a level component no members of which have neighbors larger than h. A *peak component* P_h at level h is a connected component of $X_h(f)$. At each level h there may be several such components, which will be indexed as L^i_h, P^j_h and M^k_h, respectively, with i, j, and k from some index set. It can be seen that any regional maximum M^k_h is also a peak component, but the reverse is not true. Examples of these three types of components, and of the threshold sets $X_h(f)$ are given in Figure 2.

All level components L^i_h at level h are of course subsets of some peak component $P^j_h \subset X_h(f)$ at the same level h. However, for a given criterion T, not necessarily all $L^i_h \subset \Gamma_T(P^j_h)$, because not all P^j_h need meet the criterion T. It can be seen from (3), that not all level components are necessarily affected by a grey scale attribute opening. Only those L^i_h which are *not* subsets of a peak component P^j_h which meets the criterion T must be changed in grey level by γ^T. In other words, all $L^i_h \subset \Gamma_T(P^j_h) \subset \Gamma^T(X_h(f))$ must be left unaltered. If

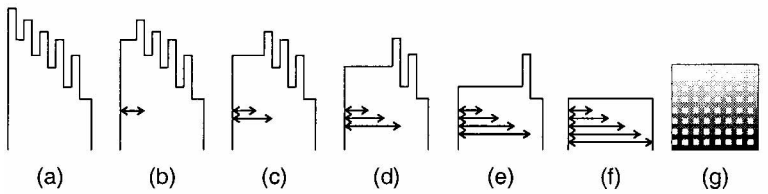

Fig. 3. Processing nested maxima by the pixel queue algorithm, assuming only the peak at full width meets the criterion: (a) original image; (b-f) situation after processing the first, second, third, fourth, and fifth maximum from the left. At each stage the pixels indicated by the double arrows have been inspected. After visiting the current maximum, the algorithm first inspects the pixels to the left of the maximum, because the valley to the right is lower. This results in a frequent rescanning of pixels of the left-most regional maxima. (g) 16 × 16 pixel image showing nested maximum structure on which the computational burden is expected to be $O(N^2 \log N)$.

we assume that the peak component $P^1_{h'}$ in Figure 2 meets the criterion, the level component labeled $L^1_{h''}$ in the same figure will remain unaltered. This is because $h'' < h'$, so that $P^1_{h'} \subset P^1_{h''}$ (the latter is not shown in the figure), and since T must be increasing in the case of a grey scale attribute opening, $P^1_{h''}$ must meet T. By contrast, assume that $P^1_h = M^1_h = L^1_h$ does not meet the criterion, and therefore $L^1_h \notin \Gamma_T(P^1_h)$ since $\Gamma_T(P^1_h) = \emptyset$. Then the grey level of L^1_h must be altered to h', because $P^1_{h'} \supset L^1_h$ is the smallest peak component containing L^1_h which meets T.

3.1. The Pixel Queue Algorithm

The pixel queue based algorithms for morphological area and attribute operators are given in some detail elsewhere [1, 5, 8, 9], so we will describe them only briefly. The source code of our implementations is available on request.

Briefly, the image is first scanned using a pixel queue to create a list of all regional maxima M^k_h. After this, all M^k_h are processed sequentially. This is done by growing a peak component $P^j_{h_1}, h_1 \leq h$ around a seed pixel within the maximum M^k_h using a priority queue. As each pixel is added to the growing region, its neighbors which do not (yet) belong to the region are put in the priority queue, from which they are retrieved in reverse grey level order. The process of adding pixels pauses whenever the next pixel taken from the priority queue has a grey level h'' different from the current level h'. If $h'' > h'$, the region grown so far is not a peak component $P^j_{h'}$ at level h'. All the grey levels of pixels found so far are set to h', and the maximum M^k_h from which the region was grown is removed from the list. If $h'' < h'$, the region grown so far is a peak component at h', which is subsequently checked against the criterion. If the criterion is met, the grey level of all pixels $p \in P^j_{h'}$ are set to h', and M^k_h is removed from the list. Otherwise, the routine continues adding new pixels at level h''. The algorithm terminates when all maxima have been processed.

One problem which occurs is that pixels may be visited more than once if nested maxima exist, especially if the attribute threshold λ is large. This effect can be seen in Figure 3. In this one-dimensional example, the algorithm processes the maxima from left to right, and each time only detects that the growing region is not a peak component *after* having re-visited all pixels visited

by the previous region-growing loop. If λ is chosen so that the entire image (of N pixels) is the smallest set satisfying the criterion, it is possible to construct an image in which each pixel is processed $O(N)$ times. A two-dimensional example can be seen in Figure 3g. At each visit a pixel has to be inserted into and retrieved from a priority queue of length of order $\sqrt{N}$. In that case we arrive at a worst case running time of $O(N^2 \log N)$. Strictly speaking, this effect is due to awkward arrangement of the depths of the valleys between the maxima, not to the heights of the maxima. Had all the maxima in figure 3(a) been given the same height (as is the case in figure 3(g)), the problem still remains. Processing the maxima in order of grey level does not solve the problem.

The algorithm requires a label image of N pixels, and a (priority) queue also of N pixels in the worst case. Therefore its memory requirements are $O(N)$.

3.2. The Union-Find Method

Tarjan [7] presents the union-find algorithm which provides a general method for keeping track of disjoint sets. It allows performing set-union operations on sets which are in some way equivalent, while ensuring that the end product of such a union is disjoint from any other set. Since connected components and level components in an image are by definition disjoint sets, the union-find algorithm lends itself to any image processing method which is defined by such image components. Dillencourt et al. [2] have shown that the union-find algorithm can be used for efficient connected component labeling of arbitrary image representations. Fiorio and Gustedt propose a similar algorithm [3], and Meijster and Roerdink [4] adapted the algorithm to level-component labeling. Since attribute openings and closings are connected filters, their operation can be defined directly in terms of connected components in the binary case and level components in the grey scale case. This means Tarjan's algorithm can be adapted to attribute openings. This is born out by the application of the algorithm to area openings [5].

Tarjan uses tree structures to represent sets. Each non-root node in a tree points to its parent, while the root is flagged in some way. Two objects x and y are members of the same set if and only if x and y are nodes of the same tree, which is equivalent to saying that they share the same root. There are four important operations.

- `Makeset (x)`: Create a new singleton set $\{x\}$.
- `FindRoot (x)`: Return the root element of the set containing x.
- `Union(x,y)`: Compute the union of the two sets containing x and y.
- `Equiv (x, y)`: determine whether x and y satisfy some equivalence criterion.

For level component labeling the algorithm becomes:

```
for pixels p do
  { MakeSet (p) ;
    for all neighbors n<p do
      if ( Equiv( n, p ) )
        Union( n, p ) ;
  }
```

Note that in this context the condition `n<p` means that `n` is a pixel which has been processed before `p`. In this case `Equiv(n,p)` is true if the image value `I[n]` equals `I[p]`. `Union` calls `FindRoot` internally to determine the root nodes of the trees containing `n` and `p`. After this scan, a second "resolving" scan assigns each root pixel a unique label, and to each non-root pixel the label of its root.

Before going into the details of the attribute opening algorithm itself, we will discuss the general framework for storing the disjoint sets, and the auxiliary functions needed for attribute openings and closings.

The disjoint sets we have to find are all level components $L_h^i \subset \Gamma^T(X_h(f))$ which are not altered by the attribute opening γ^T, and, for all other L_h^i, the smallest peak component $P_{h'}^j \supset L_h^i$ which meets T. Each of these sets is represented as a tree, with each pixel containing a pointer to its parent pixel. To store the trees for the entire image, we use an integer array `parent` of the same size as the image (i.e., N), in which `parent[p]` is the parent of pixel `p`. Pixels are stored as `width*y+x`, with `x` and `y` the pixel's x and y coordinates, and `width` the image width. If a pixel is a root of a tree, i.e. it has no parent, we flag this by setting `parent[p] < 0`. If `p` is the root of a *peak* component which does not meet the criterion, we call the pixel an *active root*, which is flagged by `parent[p] =ACTIVE< 0`. All other roots are labeled `INACTIVE` (< 0). An array `auxdata` of N void pointers is used to store pointers to any auxiliary data about the peak component (e.g., area size, centroid location, etc.) needed for computation of the attribute. Only if `p` is an active root does `auxdata[p]` point to valid data. This allows us to process different extrema simultaneously, rather than sequentially.

To perform any kind of attribute opening using a single routine, pointers to four functions must be passed to the attribute opening procedure:

- `NewAuxData`, which initializes the auxiliary data,
- `DisposeAuxData`, which discards them,
- `MergeAuxData`, which merges two sets of auxiliary data,
- `Attribute`, which computes the attribute based on the auxiliary data.

The pseudo-code for the `MakeSet`, `FindRoot`, `Equiv`, and `Union` routines is shown in Figure 4. In this case, as in the case of the area opening [5], the `Equiv` and `Union` routines are asymmetrical. This is done to ensure that if the set we are dealing with is a peak component P_h at level h, the root element `r` has a grey level `I[r]` $= h$. Therefore, we process the pixels in decreasing grey level order, and always make the last pixel processed the root of the new tree. We do this by radix-sorting the pixels, and storing the coordinates in an array `Sortpixels` of length N. Pixels of the same grey level are processed in scan line order. Scanning of peak components from high to low grey levels is guaranteed, without finding regional maxima explicitly.

As each pixel `p` is processed, we first check whether its grey level `I[p]` is different from its predecessor's greylevel `I[p-]`. If so, all active roots with grey level `I[p-] q` are inspected, checking whether `Attribute(auxdata[q])` $\geq \lambda$. If so, they are labeled as `INACTIVE` and their auxiliary data are discarded. After this clean up, the `MakeSet` routine labels `p` as a singleton set, setting `parent[p]` to `ACTIVE`, and calling `NewAuxData`, passing the pixel `p` to it (see

```
void MakeSet ( int x )
{ parent[x] = ACTIVE;
  NewAuxData(x);
}

void Link ( int x, int y )
{ if ( (parent [y] == ACTIVE) and (parent[x] == ACTIVE) )
    { auxdata[y] = MergeAuxData(auxdata[x] , auxdata[y]);
      DisposeAuxData(auxdata[x]);
    }
  else if (parent[x] = ACTIVE)
         DisposeAuxData(auxdata[x]);
       else
         { DisposeAuxData(auxdata[y]);
           parent [y] = INACTIVE; }
  parent[x] = y;
}

int FindRoot ( int x )
{ if ( parent[x] >=0 )
    { parent[x] = FindRoot( parent[x] );
      return parent[x] ;
    }
  else return x;
}

boolean Equiv ( int x, int y )
{ return ( (I[x] == I[y]) or (parent[x] == ACTIVE) );
}

void Union ( int n, int p )
{ int r = FindRoot(n);
  if  (r  !=p)
    if ( Equiv(r, p) )
      Link( r, p );
    else if (parent[p] == ACTIVE)
           { parent[p] = INACTIVE;
             DisposeAuxData(auxdata[p]);  }
}
```

Fig. 4. The basic operations for attribute openings and closings using the union-find method. The negative constants ACTIVE or INACTIVE flag active and inactive roots in the `parent` array, and `auxdata[p]` contains pointers to auxiliary data. The variable `lambda` is equal to the parameter λ. The parameters of *Equiv* must be root nodes. Linking is done if either the image values `I[x]` and `I[y]` are identical, or if `x` is a root of an active peak component.

figure 4). The `Union` procedure is now called for each neighbor `n` which has already been processed. We briefly describe this procedure here. Since the `p` is *always* a root, `FindRoot` is only called to find the root pixel `r` of `n`. Next, `Equiv` is called with `r` and `p` as parameters. If the grey level `I[r]` of `r` is equal to that of `p` *or* if `r` is active, `Equiv` returns "true" and the two trees are merged using the `Link` routine (see figure 4). If Equiv returns "false", a neighbor has a root grey level higher than `I[p]` *and* is inactive, so $\mathtt{p} \in L_h^i \subset \Gamma^T(X_h(f))$. Therefore, `p` is set to inactive, and its auxiliary data are discarded. The `Link` routine always assigns `p` to `parent[r]`. Before that, `Link` inspects both roots. If both `r` and `p` are active, `MergeAuxData` is called on the auxiliary data of `r` and `p`, storing the result in the auxiliary data of `p`, and discarding the auxiliary data of `r`. If either `r` or `p` are inactive, the active root is set to inactive, and its auxiliary data are discarded.

In pseudo code this part of the algorithm now becomes:

```
for pixels p do
  { if ( I [p] != I [p-] )
      for all pixels q with I [q]==I [p-] do
          if ( ( parent [q] == ACTIVE ) and
                ( Attribute(auxdata[q]) >= lambda ) )
             { parent [q] =INACTIVE;
               DisposeAuxData(auxdata [q] ) ;
             }
    MakeSet (P) ;
    for all neighbors n<p do
        Union, (n, p ) ;
  }
```

Here `p-` denotes the pixel processed immediately before `p`. This part of the algorithm requires $O(N \log N)$ operations in the worst case [5, 7].

At the end of this part of the algorithm, we have found two kinds of disjoint sets: (i) those with constant grey level, which are level components $L_h^i \subset \Gamma^T(X_h(f))$, and (ii) those with varying grey level, which are peak components P_h^j, with h the maximum grey value for which the criterion is satisfied. Because the root r of these peak components is always the last pixel processed, its grey level in the input image satisfies $f(r) = h$. Therefore, if we set the grey level of each pixel in the output image to that of its root in the input image, all $L_h^i \subset \Gamma^T(X_h(f))$ remain unchanged, whereas all P_h^j are filled uniformly with a grey level of h, as in the previous algorithm. Assigning all pixels the grey level of the root of their component can be done in linear time [5]. The simplest approach is to store the output image in the parent array:

```
For each pixel p in reverse sort order do
  if (parent [p] < 0) then
    parent [p] = I [p] ;
  else
    parent [p] = parent [parent [p] ] ;
```

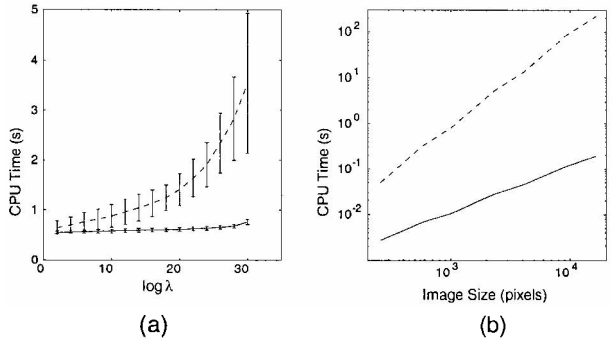

Fig. 5. Timing of algorithms for mean and pathological cases: (a) Mean CPU times and standard deviations of moment of inertia closings as a function of λ for the union-find (solid line) and pixel queue (dashed line) algorithms, for 20 natural images of 256 x 256 pixels; (b) log-log plot of timing results as a function of image size N for both algorithms, for the case shown in Figure 3a, showing at least $O(N^2)$ complexity for the pixel queue method (dashed), versus $O(N)$ for union-find (solid).

Thus, each pixel which has a negative `parent[p]` is a root, and is assigned its image value. Every image value which is not a root has a `parent[p]` which always points to a pixel which was processed *later* in the first phase of the algorithm, and which will therefore always have been assigned the correct image value *before* the current pixel in the reverse order scan. At the end of this phase, the array `parent` contains the output image.

It might be thought that finding all components $L_h^i \subset \Gamma^T(X_h(f))$ as well as those L_h^i which need to be changed is wasteful, compared to the region-growing phase of the pixel queue approach, which only grows the peak components, without visiting those L_h^i which need not be altered. However, during the phase in which the maxima are sought, the pixel queue algorithm also visits all L_h^i, regardless of whether they should be altered or not.

4. Timing Results

To compare the computational complexities of both algorithms on real images as a function of the attribute threshold λ, we computed moment of inertia openings with increasing λ for 20 natural images of 256 x 256 pixels, including microscopic images, buildings, portraits, aerial photographs and astronomical images. The results are shown in figure 5a. Similar results were obtained using the diagonal of the minimum enclosing rectangle as attribute (data not shown). As in the case of area openings and closings [5], CPU times for the pixel queue algorithm depend strongly on λ, and on image content; hence the large standard deviations of the timings. The union-find approach is faster in all cases, except for small λ in a few images. The coefficient of variance of the timings is also much smaller, indicating a far smaller dependence on image contents.

A far more dramatic difference in CPU times was observed in a set of artificial images, designed to demonstrate the $O(N^2 \log N)$ worst case behavior of the pixel queue algorithm. The images consist of maxima nested in such a

way that each time a new maximum is flooded, *all* previously processed pixels must be visited again, in a two-dimensional variant of the case shown in Figure 3. A 16 x 16 pixel image from this set is shown in figure 3g. A log-log plot of CPU times versus image size for both algorithms for images ranging from 16 x 16 up to 128 x 128 pixels is shown in figure 5b. The pixel queue algorithm shows a quadratic dependency of CPU time on image size, and at a size of 256 x 256 the CPU time was 4340 s. By contrast, the union-find algorithm shows a linear dependency on image size, and needs only 0.7 s for an image of 256 x 256 pixels. It appears that the pixel queue approach does not handle nested extrema efficiently, contrary to what has been claimed [l, 8, 9].

5. Conclusions

It has been shown that the union-find algorithm is a fast method for computing attribute openings, especially at high values of λ. Its theoretical worst case is an order of magnitude smaller than that of the pixel queue algorithm, though it may require more memory, depending on the attribute used. However, both algorithms require $O(N)$ memory in the worst case (apart from the image itself).

Further work is in progress to extend the algorithm to computation of attribute granulometries. It is expected that the speed gains will be considerable, since the pixel queue methods are particularly slow when large connected components must be scanned, i.e. at large λ. The union-find algorithm does not suffer from this drawback. A comparison to the MAX-tree approach for computation of anti-extensive connected filters [6] must also be made.

References

1. E. J. Breen and R. Jones. Attribute openings, thinnings and granulometries. *Computer Vision and Image Understanding*, 64(3):377–389, 1996.
2. M. Dillencourt, H. Samet, and M. Tamminen. A general approach to connected-component labeling for arbitrary image representations. *Journal of the ACM*, 39:253–280, 1992.
3. C. Fiorio and J. Gustedt. Two linear time union-find strategies for image processing. *Theoretical Computer Science (A)*, 154:165–181, 1996.
4. A. Meijster and J. B. T. M. Roerdink. A disjoint set algorithm for the watershed transform. In *Proc. EUSIPC0'98, IX European Signal Processing Conference*, pages 1665–1668, Rhodes, Greece, 1998.
5. A. Meijster and M. H. F. Wilkinson. An efficient algorithm for morphological area operators. Technical Report 99-9-07, Institute for Mathematics and Computing Science, University of Groningen, submitted.
6. P. Salembier, A. Oliveras, and L. Garrido. Anti-extensive connected operators for image and sequence processing. *IEEE Transactions on Image Processing*, 7:555–570, 1998.
7. R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *Journal of the ACM*, 22:215–225, 1975.
8. L. Vincent. Grayscale area openings and closings, their efficient implementation and applications. In *Proc. EURASIP Workshop on Mathematical Morphology and its Application to Signal Processing*, pages 22–27, Barcelona, Spain, 1993.
9. L. Vincent. Morphological area openings and closings for grey-scale images. In Y.-L. O, A. Toet, D. Foster, H. J. A. M. Heijmans, and P. Meer, editors, *Shape in Picture: Mathematical Description of Shape in Grey-level Images*, pages 197–208. NATO, 1993.